# Parallel partitioning method (PPM): A new exact method to solve bi-objective problems

Julien Lemesre, Clarisse Dhaenens, El-Ghazali Talbi

# Parallel Partitioning Method (PPM): A new exact method to solve Bi-Objective problems.

J. Lemesre [a,*], C. Dhaenens [a], E.G. Talbi [a]

[a] *LIFL, Bâtiment M3, Université de Lille I,*

*59655 Villeneuve d'Ascq cedex, France*

**Abstract**

In this paper, we propose a new exact method, called the Parallel Partitioning Method (PPM), able to solve efficiently bi-objective problems. This method is based on the splitting of the search space into several areas leading to elementary exact searches. We compare this method with the well known Two-Phase Method (TPM). Experiments are carried out on a Bi-Objective permutation FlowShop Problem (BOFSP). During experiments the proposed PPM is compared with two versions of TPM: the basic TPM and an improved TPM dedicated to scheduling problems. Experiments show the efficiency of the new proposed method.

*Key words:* Exact method, bi-objective problem, permutation flowshop.

## Introduction

A large part of real-world optimization problems are of multi-objective nature. In trying to solve Multi-objective Combinatorial Optimization Problems

* corresponding author. Tel.: +33 3 20 41 75 63. Fax.: +33 3 28 77 85 37.
*Email addresses:* lemesre@lifl.fr (J. Lemesre), dhaenens@lifl.fr (C. Dhaenens), talbi@lifl.fr (E.G. Talbi).

(MOPs), many methods scalarize the objective vector into a single objective. However, since several years, interest concerning MOPs area with Pareto approaches grows.

MOPs are most of the time NP-hard, and this difficulty is enforced by the existence of several optimal solutions. Hence, a lot of heuristic methods have been proposed to solve multi-objective (and bi-objective) problems. In this paper, we are interested in exact resolutions of MOPs and we propose a new exact method able to solve bi-objective problems, named the Parallel Partitioning Method. This method is able to enumerate all the Pareto solutions of a bi-objective problem.

Some exact methods have already been proposed to solve bi-objective combinatorial optimization problems. To find details on some of them the reader may refer to [1,2]. A very well known approach is the weighted sum scalarization. But, as we will see later, this approach is not able to produce the whole Pareto set. Other exact approaches are the application of the $\epsilon$-constraint concept proposed by Haimes et al. [3,4], or the Two-Phase Method, proposed by Ulungu and Teghem [5]. These methods have some disadvantages: for example, in the $\epsilon$-constraint method, one search is required to get each solution belonging to the front which can be very long for problems having a lot of Pareto solutions. Regarding the Two-Phase Method, let us remark that its success is based on the good distribution of supported solutions along the Pareto front and on the efficiency of mono-objective searches. In order to overpass these drawbacks, we propose a new method, PPM, which integrates the advantages of the two previously cited methods.

In this paper, minimization problems are addressed.

This paper is organized as follows: Section 1 defines Multi-Objective Combinatorial Optimization. Section 2 presents the bi-objective permutation flow-

shop problem used as an illustration. Section 3 presents existing exact methods and in particular the $\epsilon$-constraint method and the original Two-Phase Method (TPM). Section 4 describes PPM, the new proposed Parallel Partitioning Method and compare PPM and TPM. Section 5 gives comparative performance results of these approaches. Finally, we expose conclusions and perspectives of this work.

## 1 Multi-objective Optimization

Before presenting the bi-objective problem used as an example, we describe and define Multi-objective combinatorial Optimization Problems (MOPs) in the general case in order to introduce notations used in this article.
In multi-objective problems, where an objective vector ($f_1$, $f_2$,...,$f_p$) has to be optimized, there exists not a single optimal solution but a set of solutions of best compromise. These solutions are forming the Pareto set. They may be defined thanks to the dominance notion.

**Definition 1** *In a minimization problem, a solution x dominates a solution x' if and only if:*

$$\begin{cases} \forall k \in [1..p], f_k(x) \leq f_k(x') \\ \\ \exists k \in [1..p], f_k(x) < f_k(x') \end{cases}$$

Therefore the Pareto optimality definition is:

**Definition 2** *A solution is Pareto optimal if it is not dominated by any other solution of the feasible set.*

In this paper we are interested in developing a new exact method (PPM) able to enumerate all the Pareto solutions for bi-objective problems.

Hence, we compare the Two-Phase Method (TPM) with the Parallel Partitioning Method (PPM). To compare these methods, the problem used is a bi-objective permutation flowshop.

## 2 A Bi-Objective permutation FlowShop Problem (BOFSP)

The flowshop problem is one of the numerous scheduling problems. It has been widely studied in the literature (for example see [6–8] for resolution of mono-objective permutation flowshop problems). A survey of the existing multi-objective approaches for scheduling problems may be found in [9,2].
The flowshop problem consists in scheduling $n$ jobs ($i = 1 \dots n$) on $m$ machines ($j = 1 \dots m$). In this work, we study the permutation flowshop where the jobs are scheduled in the same order on all the machines.
The two considered objectives are the makespan, ($C_{max}$) and the total tardiness ($T$). The makespan is the completion time of the last job and the total tardiness is the sum of tardinesses of every job. In the Graham et al. notation [10], this problem is denoted $F/permut, d_i/(C_{max}, T)$.
The makespan minimization problem has been proved to be strongly NP-hard by Garey, Johnson and Sethi [11] for permutation flowshops with more than two machines whereas the total tardiness minimization problem has been proved to be NP-hard by Du and Leung [12] even on a single machine.
Before presenting PPM, two existing exact methods are presented.

## 3 Existing exact methods

A well known method to solve multi-objective problems is the weighted sum method, but this method is not able to find the whole Pareto set. There exist several other exact methods, whose goal is to obtain the whole Pareto set : the dichotomic Method, the $\epsilon$-constraint Method and the Two-Phase Method (TPM) (see [1] for more details). Here, two methods, TPM and the $\epsilon$-constraint method, are presented in details, since PPM will integrate the advantages of these two methods. An improved TPM dedicated for scheduling problems is also exposed (see [13] for more details). Methods in this section are presented for the bi-objective case.

### *3.1 The $\epsilon$-constraint Method*

The $\epsilon$-constraint Method is an application of the $\epsilon$-constraint concept (introduced in [3,4]) to enumerate Pareto solutions. This method involves a constraint on one objective and optimizes the second objective. The constrained problem may be expressed as follows: $\min \{f_1(x) : x \in \mathcal{X},\ with\ f_2(x) \leq\ \epsilon\}$. First, one extreme is computed, for example the extreme with the best value on the objective $f_1$. This solution gives a bound on the objective $f_2$ and the best solution regarding to objective $f_1$ is searched below this bound (see Fig. 1). This operation is repeated (see Fig. 2) until no new solution is found. Hence all the efficient set is known.

*Initial Method*

In 1995, Ulungu and Teghem [5] proposed the Two-Phase Method to initially solve a bi-objective assignment problem. This method proposes a very general scheme that can be adapted to specific problems. Hence, an adaptation of the Two-Phase Method for the bi-objective permutation flowshop has been presented [13].

The first phase consists in finding all the supported solutions with aggregations in the form $\lambda_1 f_1 + \lambda_2 f_2$ by recursively exploring the existence of a supported solution "between" (in the objective space) two given supported solutions. Each time a solution is found, two new searches are launched (see Fig. 3). Therefore only supported solutions (solutions that belong to the convex hull) are found.

Once all the supported solutions have been found, the second phase consists in exploring all the triangles, underlying each pair of adjacent supported solutions, in order to find all the non-supported solutions (Fig. 4).

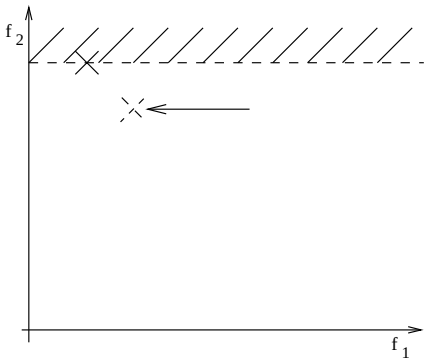


Fig. 1. One example of search in the $\epsilon$-constraint method.

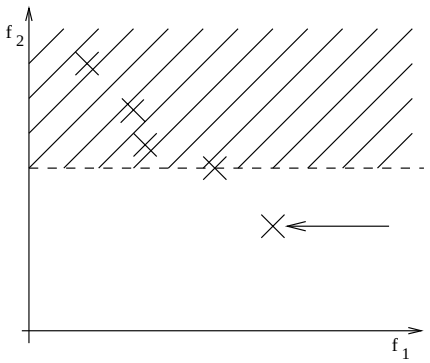


Fig. 2. New search.

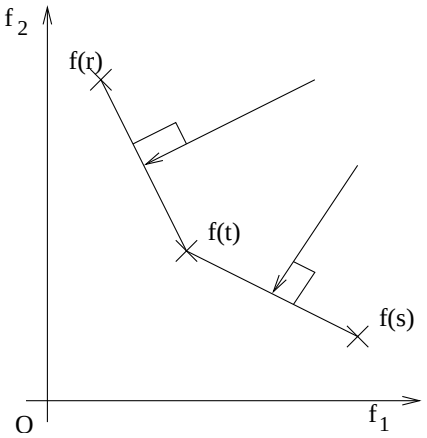


Fig. 3. Search direction.

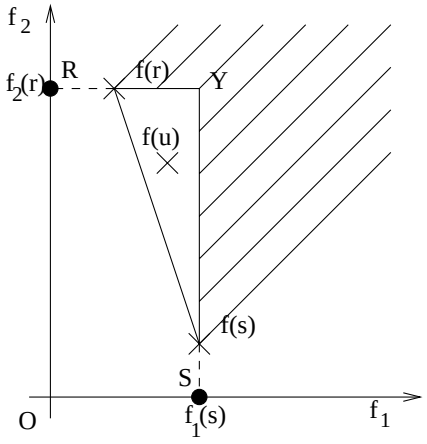


Fig. 4. Location of non-supported solutions.

*Characteristics of the TPM and improvements*

With the Two-Phase Method, a bi-objective problem is solved exactly without exploring the whole search space. Only the pertinent search space is visited. In problems solved by Ulungu and Teghem [5,14] each single objective problem may be solved efficiently thanks to dedicated methods and aggregations of these objectives as well. This explains the efficiency of TPM on such problems, since TPM uses these dedicated methods to solve sub-problems. For example, the Hungarian methodis able to solve efficiently the assignment problem and is used by TPM to find extremes and supported solutions [5]. In our adaptation to BOFSP, we use a Branch & Bound method to solve aggregations as there exists no efficient method (polynomial method) for the flowshop problem associated with the studied objectives (see [13] to have more details on this Branch & Bound). Hence, for the BOFSP, the first phase can be time consuming.

In addition, the efficiency of the TPM is also based on the size of the generated triangles which depends on the number of supported solutions in the Pareto set and on their distribution along the front. In the permutation flowshop,

supported solutions may be vary close. A result of one instance with 20 jobs and 10 machines is represented on Figure 5. In this experiment, we can see that 5 supported solutions are adjacent (on the top of the front). Hence, for this part of the front, the TPM will not be very efficient.

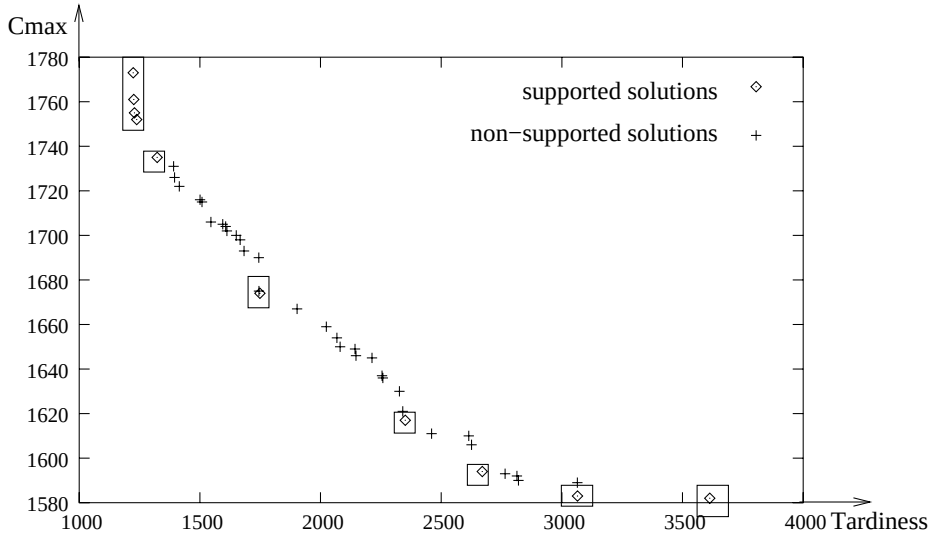


Fig. 5. Result 20 jobs, 10 machines, No1.

In response to these remarks, we proposed some improvements (see [13] for more details) to speed up the method when it is applied to scheduling problems. The first improvement deals with the computation of the extremes. The second improvement aims to avoid useless searches during the first phase.
Using these two improvements, we managed to solve problems faster (see Table 1 and Table 2). But during the first phase of TPM only supported solutions are found. So, the search space obtained for the second phase may be very large if supported solutions are not well distributed along the front. Moreover, the search of supported solutions can be very long when the problem has no interesting structure. For these reasons, we propose a new method that keeps the idea of the space partitioning but manages it in a different way.

## 4 The Parallel Partitioning Method (PPM)

The new proposed PPM is a method able to solve any bi-objective problem. This method provides a general scheme in order to find the whole Pareto set in three stages.

### *4.1 First stage: search of the extremes*

The first stage consists in finding the two extreme efficient solutions which are Pareto optimal. In order to speed up this search, we propose to use a lexicographical order on the objectives. Hence, for example, to obtain the extreme Pareto solution for objective $f_1$, $f_1$ is first optimized without optimizing $f_2$. Secondly the best value for $f_1$ is kept and $f_2$ is optimized.
These two solutions give bounds on the pertinent search space because they indicate the lowest and the largest possible values on each objective for any Pareto optimal solution.

### *4.2 Second stage: split of the search space*

The objective of this stage is to find a subset of well distributed Pareto solutions. Therefore the second stage consists in equally splitting the search space. Extreme solutions found at the first stage are used to make the split. This splitting is done according to one objective. In Figure 6, four splits have been made according to objective $f_2$.
Then, for each split a specific Pareto solution is computed: the one that respects the considered split and has the best value for the objective not used for

the splitting. Therefore, one objective is limited and the second is optimized. This process is similar to the $\epsilon$-constraint method [3].

Solutions found for splits of Figure 6 are shown on Figure 7. It is important to notice that this stage finds supported and non-supported solutions. So, if the distribution of the Pareto optimal solutions is well sparse, then the obtained subset of Pareto solutions is also well sparse.

The objective chosen to make the split does not change the performance of the method. The two schemes have been tested for the studied BOFSP and times required to solve instances are equivalent.

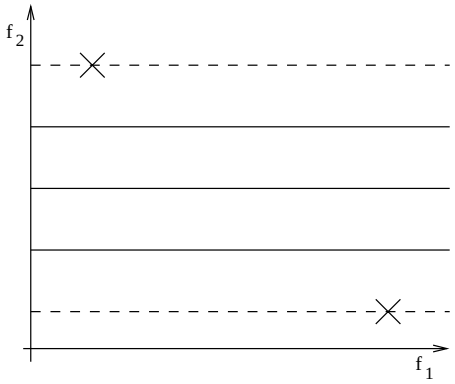


Fig. 6. Split of search space.

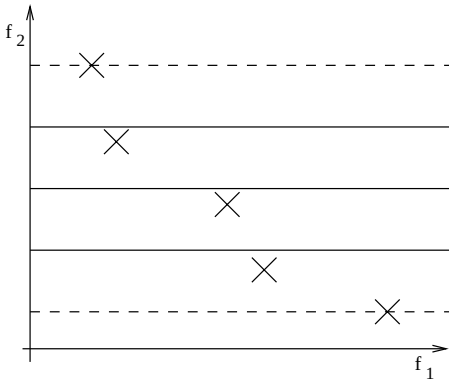


Fig. 7. Search of solutions well distributed on the Pareto front.

### *4.3 Third stage: search of all the other Pareto solutions*

The third stage consists in finding all the Pareto solutions not found during the previous stages. This stage is similar to the second phase of the improved TPM. For two adjacent solutions $f(s)$ and $f(r)$ the search is made in the rectangle $SYRO$ where $Y$ is the point $(f_1(s),\ f_2(r))$, $R$ is the point $(0, f_2(r))$ and $S$ is the point $(f_1(s), 0)$ (see Fig. 8). In fact, this search can be limited to the rectangle $f(r)Yf(s)Z$ where $Z$ is the point $(f_2(s),\ f_1(r))$, if the search

method is able to handle such constraints.

The whole search space explored at this stage is represented in Figure 9. No solution can dominate the extremes (grey space in the figure). Moreover if one solution is not in this search space (if this solution is in the hatched space), this solution is dominated by one or more previously found solutions. To search

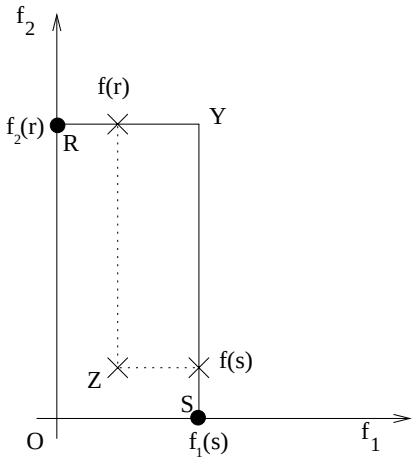


Fig. 8. Rectangle of search.

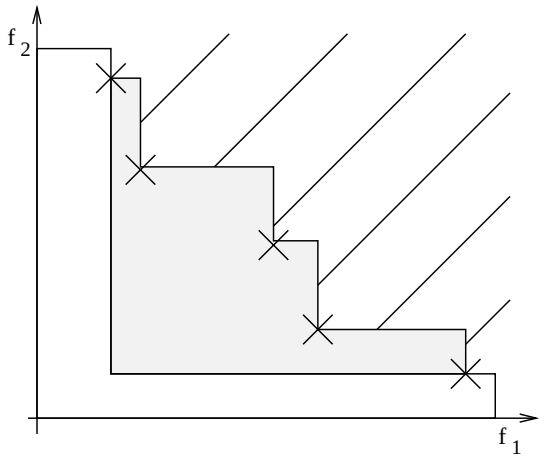


Fig. 9. Search space for third stage.

inside the rectangles, search directions can be defined by $\lambda_1$ and $\lambda_2$ computed as in TPM ($\lambda_1 = f_2(r) - f_2(s)$, $\lambda_2 = f_1(s) - f_1(r)$).

## 5 Comparisons of PPM and TPM

In this section, we discuss the particularities of TPM and PPM and their advantages and disadvantages.

### *5.1 Solutions used for the split*

In order to split the search space, TPM proposes to use all the supported solutions, whereas PPM uses well distributed Pareto (supported or non sup-

ported) solutions. Hence, in TPM, lot's of searches are made to find all the supported solutions and some of them do not produce any new solution but are used to prove that no new supported solution exists. Therefore the number of mono-objective searches can be high in TPM. When mono-objective searches required to find supported solutions are polynomial (case of well structured problems), finding all the supported solutions is not time consuming. But when these searches are NP-Hard, it is interesting to limit their number. This is what PPM does.

### *5.2 Balance the search space*

In TPM, the size of the triangles obtained for the second phase depends on the distribution of supported solutions and may be very unbalanced (very small and very large triangles). As far as PPM is concerned, the size of searches for the third stage are dependent on supported and non-supported solutions, that is to say on the distribution of the whole set of solutions of the Pareto front. Figure 10 illustrates this fact. The search space defined by TPM and the search

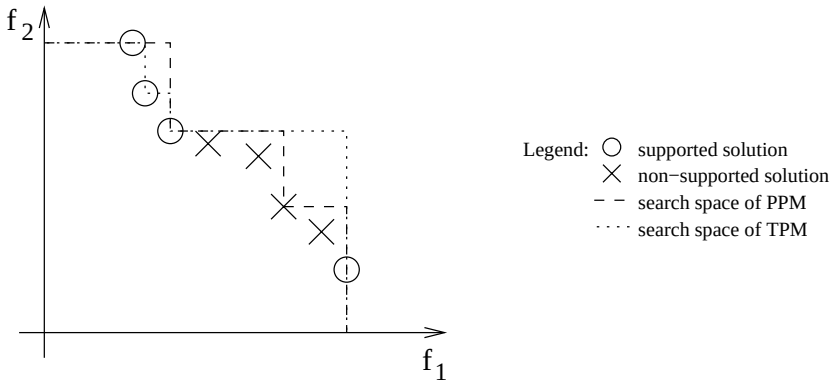


Fig. 10. Search spaces of the methods.

space defined by PPM are presented. On this figure, supported solutions are indicated with rounds and non-supported with crosses. In this example, seven

mono-objective searches are required by TPM to find the extremes and all the supported solutions (and to prove that no other supported solution exists). With PPM, only four searches are required to find the extremes and the two solutions used for the split. Moreover the search sub-spaces obtained with PPM are well balanced than the search sub-spaces obtained with TPM because in PPM, supported and non-supported solutions can be used to delimit this search space.

### *5.3 Number of partitions*

TPM and PPM are very general schemes that could be applied to any bi-objective problem. In PPM, the number of partitions has to be determined. This is a parameter of the procedure. This number must be determined according to the studied problem and the knowledge we may have on the form of the Pareto set. In TPM the first phase, used to reduce the search space, can be very time consuming, especially when the mono-objective searches are NP-hard. In PPM, the determination of the number of partitions give control on the time required to find solutions used for the split.

### *5.4 Structure of problems*

During the second step of PPM, a constraint is added on one objective. With this constraint, the structure of the original problem may be broken and existing efficient method dedicated for this problem may becaume unusable. Hence, for these problems, TPM will be efficient. On contrary, for NP-hard mono-objective problems, like in our case the BOFSP, exact mono-objective

methods do not use the structure of problems. Therefore for these problems PPM will be efficient.

The discussion about these four points show that in theory, PPM should be more efficient than TPM in particular on a difficult problem such as the BOFSP. These conclusions have to be verified thanks to experiments.

## 6 Experiments

To evaluate the different methods on the BOFSP presented in section 2, we use benchmarks proposed by Taillard [15] and by Reeves [16] for the problem $F/permut/C_{max}$ , to which due dates for jobs have been added [1] in order to be able to compute the total tardiness objective.
Instances are described with the origin of the instance, the number of jobs, the number of machines and the number of the instance. For example, the instance *taill_20_5_1* is the first instance of Taillard with 20 jobs and 5 machines.
All these experiments have been carried out on a computer with a Pentium IV microprocessor of 3 GHz. In this part, methods are compared and different aspects are discussed.
In order to compare the different methods, same approaches, based on Branch & Bound, are used to solve sub-problems.

[1] details on this extension may be found at *www.lifl.fr/∼lemesre*

In this section, we compare methods while looking for the minimal set (only solutions with different values of objectives are searched). For the PPM method the split is made according to the total tardiness criterion. Previous experiments have shown that the objective chosen to split, has no influence on the time required to solve instances on BOFSP. Table 1 presents results obtained with different methods for Taillard's Benchmarks. This table indicates the number of solutions (supported and non-supported) of the front and the time required to solve the instances for three different methods: the original Two-Phase Method (TPM), the improved Two-Phase Method and the Parallel Partitioning Method (PPM). It shows that improvements on TPM allow an important reduction of the search time, in particular for problems of size $20 * 10$. It shows also that the search time for PPM is even smaller than for the improved TPM.

Table 1

Taillard's benchmarks: Times required to obtain the minimal Pareto set.

| Instances | Number of solutions | | Time | | |
|---|---|---|---|---|---|
| | Supported | Non Supported | Original TPM | Improved TPM | PPM |
| taill_20_5_1 | 3 | 1 | 6.5 s | 3.5 s | 3.3 s |
| taill_20_5_2 | 2 | 4 | 3 mn 10 s | 2 mn 50 s | 2 mn 50 s |
| taill_20_10_1 | 10 | 32 | 1 day 3 h | 12 h 25 mn | 8 h 58 mn |
| taill_20_10_2 | 8 | 23 | 18 h 04 mn | 7 h 23 mn | 5 h 01 mn |

Table 2 presents similar results obtained for benchmarks proposed by Reeves

(*reC_nbJob_nbMach_nb*). For these benchmarks, we produce two types of instances; "easy" and "hard", according to the difficulty to optimize the total tardiness in the mono-objective case. For these two types the same information as in Table 1 is given.

Table 2

Reeves' Benchmarks: Times required to obtain the minimal Pareto set.

| Instances | Number of solutions | | Time | | |
|---|---|---|---|---|---|
| | Supported | Non Supported | Original TPM | Improved TPM | PPM |
| | | | Easy | | |
| reC_20_5_1 | 7 | 17 | 53 s | 20 s | 18 s |
| reC_20_10_7 | 3 | 3 | 3 mn 01 s | 35 s | 31 s |
| reC_20_15_13 | 4 | 7 | 5 h 10 mn | 2 h 10 mn | 1 h 55 mn |
| reC_20_15_17 | 5 | 14 | 10 h 41 mn | 6 h 01 mn | 3 h 42 mn |
| | | | Hard | | |
| reC_20_5_1 | 8 | 34 | 54 s | 39 s | 32 s |
| reC_20_10_7 | 9 | 31 | 1 h 22 mn | 32 mn 41 s | 22 mn 19 s |

This table allows to make similar conclusions as for Table 1: improvements on TPM allow a reduction of the search time and the time required by PPM is even smaller than for the improved TPM. Moreover we can see that the time required to solve a given size of problems depends on the difficulty of the total tardiness associated problem. In particular, problems of size $20 * 15$ of yhe hard class problem can not be solved within one week whereas problems

of the same size of the easy class may be solved within few hours.

### *6.2 Graphical representation of fronts*

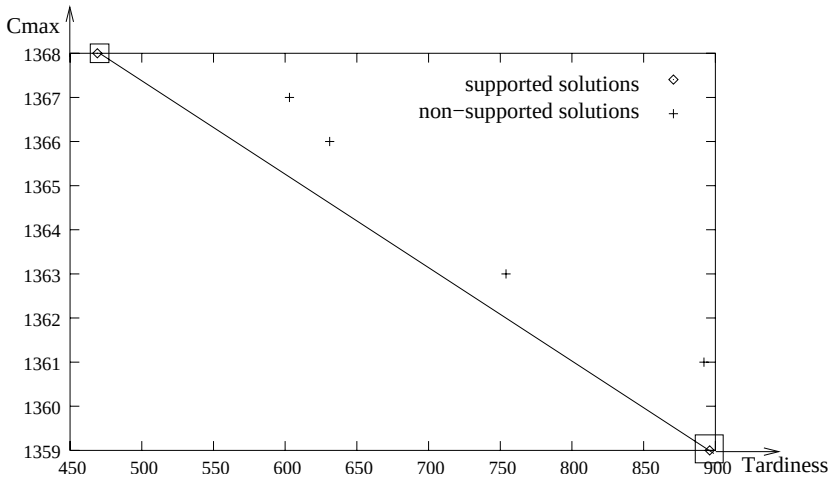


Fig. 11. Result of problem taill_20_5_2.

Figures 5 (in Section 3.2), 12 and 11 present three different Pareto fronts for three different instances. Those figures show that Pareto sets may have different structures (number of solution, number of supported/non-supported solution, distribution of the supported solutions...). For example, Figure 5 and 12 shows that supported solutions may be very close (and not well distributed). Moreover, Figure 11 shows that there may exist few supported solutions. In this case TPM is unable to split interestingly the search space. Therefore, structures of these fronts explain the good saving of time obtained with PPM.

### *6.3 Impact of the search of the extremes*

The search of the two extremes has to be made whatever the method used. Unfortunately these searches may be time consuming. Therefore to appreciate the saving of time of PPM compared to the improved TPM, these two methods

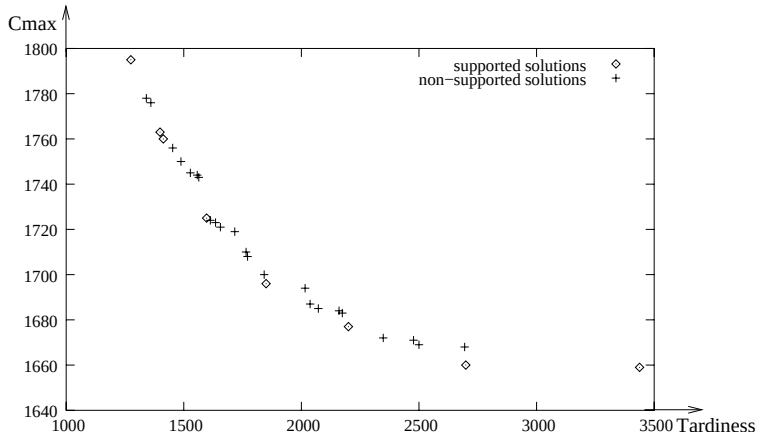


Fig. 12. Pareto set taill_20_10_2

are compared without considering the time required to search the extremes. Table 3 indicates the required time to compute the two extremes and the time required to solve the different instances once the two extremes are computed. Table 3 shows that the part of the time spent to search the extremes can represent a large part of the total time required by a given method. Moreover this table shows that without considering this common part of the work, PPM appears even more efficient than Improved TPM. Indeed, PPM can required only half of the time of TPM ($reC\_20\_15\_17$).

### *6.4 Minimal Pareto set/Maximal Pareto set*

Let us bring to the attention of the reader that depending on the space we are considering (decision space/objective space), the number of Pareto solutions may be different. In the objective space, two solutions having the same objective vector will be considered as a single point, whereas they represent two different solutions of the decision space. The Pareto set in the objective space is called *the minimal complete Pareto set* whereas it is *the maximal complete Pareto set* in the decision space. These two problems are slightly

different and it is worth precising which problem is addressed while studying a multi-objective problem.

Table 4 presents the time required by PPM to obtain the maximal complete Pareto set and the number of Pareto solutions composing the maximal com-

Table 3

Search of the extremes.

| Instances | Time | | |
|---|---|---|---|
| | Search of extremes | Improved TPM without extremes | PPM without extremes |
| | Taillard | | |
| taill_20_5_1 | 2.6 s | 0.9 s | 0.7 s |
| taill_20_5_2 | 2 mn 20 s | 30 s | 30 s |
| taill_20_10_1 | 1 h 02 mn | 11 h 23 mn | 7 h 56 mn |
| taill_20_10_2 | 1 h 05 mn | 6 h 18 mn | 3 h 56 mn |
| | Reeves Easy | | |
| reC_20_5_1 | 6 s | 14 s | 12 s |
| reC_20_10_7 | 18 s | 17 s | 13 s |
| reC_20_15_13 | 1 h 13 mn | 57 mn | 42 mn |
| reC_20_15_17 | 1 h 21 mn | 4 h 40 mn | 2 h 21 mn |
| | Reeves Hard | | |
| reC_20_5_1 | 2 s | 37 s | 30 s |
| reC_20_10_7 | 2 mn 01 s | 30 mn 40 s | 20 mn 18 s |

Table 4

Maximal Pareto set.

| Instances | Maximal set | | Minimal set | |
|---|---|---|---|---|
| | Time | # of solutions | Time | # of solutions |
| | Taillard | | | |
| taill_20_5_1 | 4 s | 18 | 3.3 s | 4 |
| taill_20_5_2 | 3 mn 15 s | 21 | 2 mn 50 s | 6 |
| taill_20_10_1 | 9 h 07 mn | 43 | 8 h 58 mn | 42 |
| taill_20_10_2 | 5 h 15 mn | 31 | 5 h 01 mn | 31 |
| | Reeves Easy | | | |
| reC_20_5_1 | 19 s | 50 | 18 s | 24 |
| reC_20_10_7 | 40 s | 13 | 31 s | 6 |
| reC_20_15_13 | 1 h 59 mn | 12 | 1 h 55 mn | 11 |
| reC_20_15_17 | 3 h 55 mn | 28 | 3 h 42 mn | 19 |
| | Reeves Hard | | | |
| reC_20_5_1 | 33 s | 71 | 32 s | 42 |
| reC_20_10_7 | 25 mn 24 s | 368 | 22 mn 19 s | 40 |

plete Pareto set. Table 4 also recalls the time required to obtain the minimal complete Pareto set and the corresponding number of solutions, in order to make a comparison.

This table shows that the time required to obtain the maximal Pareto set is comparable with the time needed to obtain the minimal Pareto set, even if for some instances, the number of solutions may be larger. Hence, the efficiency

of the method does not decrease with the search of the maximal Pareto set.

## 7 Conclusion and perspectives

In this paper, we propose a new exact method named Parallel Partitioning Method for solving bi-objective problems. This method is compared with the well known Two-Phase Method for which several improvements were proposed for scheduling problems. These methods are general scheme able to enumerate all the Pareto solutions for any bi-objective problem. PPM is a method which takes advantages of other exact methods to find Pareto optimal solutions more rapidly. It proposes an interesting split of the search space, by finding well distributed solutions.

We have chosen to validate and compare the two methods (TPM and PPM) on a bi-objective permutation flowshop problem. We present some advantages of the Parallel Partitioning Method and some particularities of the studied problem which explain the good results obtained with this method.

Further researches on this topic will deal with comparing TPM and PPM on other bi-objective scheduling and non-scheduling problems. The Parallel Partitioning Method may also be used in cooperation schemes with (meta-)-heuristics (Basseur et al. [17]). Another further research direction is to develop a parallel implementation of PPM, which should be very efficient, since the concept of the method is inherently parallel. Finally an interesting point is the extension of such a method to more than two objectives.

## References


[1] M. Ehrgott, X. Gandibleux, A survey and annoted bibliography of multiobjective combinatorial optimization, OR Spektrum 22 (4) (2000) 425–460.

[2] V. T'Kindt, J.-C. Billaut, Multicriteria Scheduling: theory, models and algorithms, Springer-Verlag, 2002.

[3] Y. Haimes, L. Ladson, D. Wismer, On a bicriterion formulation of the problems of integrated system identification and system optimization, IEEE Transaction on system, Man and Cybernetics 1 (1971) 296–297.

[4] Y. Haimes, W. Hall, H. Freedman, Multiobjective Optimization in Water Resource Systems: The Surrogate Worth Trade Off Method, Elsevier Scientific Publishing edition, New York, 1975.

[5] E. Ulungu, J. Teghem, The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems, Foundations of Computing and Decision Sciences 20 (2) (1995) 149–165.

[6] S. K. Iyer, B. Saxena, Improved genetic algorithm for the permutation flowshop scheduling problem, Computers and Operations Research 31 (4) (2004) 593–606.

[7] J. M. Framinan, R. Leisten, R. Ruiz-Usano, Comparison of heuristics for flowtime minimisation in permutation flowshops, Computers and Operations Research 32 (5) (2005) 1237–1254.

[8] J. Fondrevelle, A. Oulamara, M. Portmann, Permutation flowshop scheduling problems with maximal and minimal time lags, Computers and Operations Research. To appear.

[9] A. Nagar, J. Haddock, S. Heragu, Multiple and bicriteria scheduling: A litterature survey, European Journal of Operational Research 81 (1995) 88–104.

[10] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, in: Annals of Discrete Mathematics, Vol. 5, 1979, pp. 287–326.

[11] M. Garey, D. Johnson, R. Sethi, The complexity of flow-shop and job-shop scheduling, Mathematics of Operations Research 1 (1976) 117–129.

[12] J. Du, J. Y.-T. Leung, Minimizing total tardiness on one machine is NP-hard, Mathematics of Operations Research 15 (1990) 483–495.

[13] J. Lemesre, C. Dhaenens, E. Talbi, An exact parallel method for a bi-objective permutation flowshop problem, European Journal of Operational Research. To appear.

[14] M. Visée, J. Teghem, M. Pirlot, E. Ulungu, Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem, Journal of Global Optimization 12 (1998) 139–155.

[15] E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research 64 (1993) 278–285.

[16] C. Reeves, A genetic algorithm for flowshop sequencing, Computers and Operations Research 22 (1995) 5–13.

[17] M. Basseur, J. Lemesre, C. Dhaenens, E.-G. Talbi, Cooperation between branch and bound and evolutionary approaches to solve a biobjective flow shop problem, in: Workshop on Experimental and Efficient Algorithms (WEA'04), Rio de Janeiro, Brasil, 2004, pp. 72–86.